STATUS OF THE CLAIMS

Claims 1-20 were originally filed in this patent application. In the pending office action, claims 1-20 were rejected under 35 U.S.C. §102(b) as being anticipated by WO Patent No. 98/02813 to De Borst *et al.* (hereinafter "De Borst"). No claim was allowed. In this amendment, claims 1, 7, 8, 11, 13, 15 and 16 have been amended. Claims 1-20 are currently pending.

REMARKS

Rejection of claims 1-20 under 35 U.S.C. §102(b) as being anticipated by De Borst

The Examiner rejected claims 1-20 under 35 U.S.C. §102(b) as being anticipated by De Borst. Each of these claims is addressed below.

Claim 16

The Examiner first addressed claim 16. In the rejection, the Examiner did not address all of the limitations in claim 16. For example, the Examiner did not address the limitation that the datastream class corresponds to a received identifier. In addition, the Examiner did not address the limitation that the datastream receive mechanism "populates the instance of the datastream with received data." Because the Examiner has not addressed these limitations, the Examiner has failed to establish a prima facie case of anticipation for claim 16 under 35 U.S.C. §102(b).

Claim 16 is amended herein to specifically recite that the identifier is an identifier in a received datastream, and to recite the datastream receive mechanism populating the instance of the datastream class with information contained in the received datastream.

De Borst discloses a factory interface and a stream object. However, the factory interface and stream object in De Borst do not meet the limitations of the datastream factory and datastream receive mechanism in claim 16. The factory interface in De Borst is used to create a Stream object that is used to deliver large amounts of data between components. See De Borst, p. 11, lines 13-14. The factory interface in De Borst creates an instance of the Stream class, but does not create an instance of a class corresponding to an identifier in a received datastream. For this reason, the factory interface in De Borst does not read on the datastream factory in claim 16.

The Stream object in De Borst is an object that contains a datastream, and that includes methods for storing data to the datastream and for reading data from the datastream. Data can be stored to the Stream object in De Borst by other components invoking the put() method on the Stream object, which stores data passed with the put() call into the Stream object. Thus, in De Borst, it is the other components that populate the Stream object. Nowhere does De Borst teach that a Stream object can *populate itself* with information contained in a received datastream by invoking at least one object method on the instance, as recited in claim 16. For this reason, the Stream object in De Borst does not read on the datastream receive mechanism in claim 16. Applicants assert that claim 16 is allowable over De Borst, and respectfully request reconsideration of the Examiner's rejection of claim 16.

Claims 17-20

Claims 17-20 depend on claim 16, which is allowable for the reasons given above. As a result, claims 17-20 are allowable as depending on an allowable independent claim.

Claim 1

In rejecting claim 1, the Examiner refers to the rejection of claim 16, then states that claim 1 additionally includes a processor. Applicants respectfully assert that the Examiner has failed to address all of the limitations in claim 1. For example, claim 1 recites an identifier received from a second computer system. Claim 1 also recites the datastream receive mechanism populating the instance of the datastream class with data received from the second computer system. Because the Examiner did not address these limitations in claim 1, the Examiner has failed to establish a prima facie case of anticipation for claim 1 under 35 U.S.C. §102(b).

Claim 1 is amended herein to specifically recite that the identifier is an identifier in a datastream received from a second computer system, and to recite the datastream receive mechanism populating the instance of the datastream class with information contained in the datastream received from the second computer system. The factory interface in De Borst creates an instance of the Stream class, but does not create an instance of a class corresponding to an identifier in a datastream received from a second computer system. For this reason, the factory interface in De Borst does not read on the datastream factory in claim 1. Furthermore, nowhere does De Borst teach that a Stream object can populate itself with information contained in a datastream received from a second computer system by invoking at least one object method on the instance, as recited in claim 1. For this reason, the Stream object in De Borst does not read on the datastream receive mechanism in claim 1. Applicants assert that claim 1 is allowable over De Borst, and respectfully request reconsideration of the Examiner's rejection of claim 1.

Claims 2-6

Claims 2-6 depend on claim 1, which is allowable for the reasons given above. As a result, claims 2-6 are allowable as depending on an allowable independent claim.

Claim 7

In rejecting claim 7, the Examiner states that claim 7 is the same as claim 1 except claim 7 recites first and second computers and an active datastream, then cites to De Borst as allegedly teaching these limitations. Applicants respectfully assert that the Examiner has not addressed all of the limitations in claim 7. For example, the Examiner has not addressed the limitation that the datastream class corresponds to a datastream identifier received on the network connection from the other computer system. Furthermore, the Examiner has not addressed the limitation that the datastream receive mechanism "populates the instance of the active datastream class with data received on the network connection from the other computer system." Because the Examiner has failed to address these limitations in claim 7, the Examiner has failed to establish a prima facie case of anticipation for claim 7 under 35 U.S.C. §102(b).

Claim 7 is amended herein to specifically recite that the identifier is an identifier in a datastream received in a datastream on the network connection from the other computer system, and to recite the datastream receive mechanism populates the instance of the active datastream class with information contained in the datastream received on the network connection from the other computer system. As stated above with respect to claims 1 and 16, the factory interface in De Borst creates an instance of the Stream class, but does not create an instance of a class corresponding to an identifier in a datastream received from a second computer system. For this reason, the factory interface in De Borst does not read on the datastream factory in claim 7. Furthermore, nowhere does De Borst teach that a Stream object can populate itself with information contained in a datastream received from a second computer system by invoking at least one object method on the instance, as recited in claim 7. For this reason, the Stream object in De Borst does not read on the datastream receive mechanism in claim 7. Applicants assert that claim 7 is allowable over De Borst, and respectfully request reconsideration of the Examiner's rejection of claim 7.

Claim 8

In rejecting claim 8, the Examiner states that claim 8 is the same as claim 1 except claim 8 additionally includes a network connection. Applicants respectfully assert that the Examiner has not addressed all of the limitations in claim 8. For example, the Examiner has not addressed the limitation of the "means for constructing an active datastream" recited in claim 8. In addition, the Examiner has not addressed the limitation that the datastream class corresponds to a datastream identifier. Furthermore, the Examiner has not addressed the limitation of the "means for populating the instance of the datastream class with the active datastream received from the first computer system." Because the Examiner has failed to address these limitations in claim 8, the Examiner has failed to establish a prima facie case of anticipation for claim 8 under 35 U.S.C. §102(b).

Claim 8 is amended herein to specifically recite "means for populating the instance of the datastream class with information contained in the active datastream received from the first computer system." As stated above with respect to claims 1, 7 and 16, nowhere does De Borst teach that a Stream object can populate itself with information contained in a datastream received from a different computer system by invoking at least one object method on the instance, as recited in claim 8. For this reason, the Stream object in De Borst does not read on the means for populating the instance of the datastream class in claim 8. Applicants assert that claim 8 is allowable over De Borst, and respectfully request reconsideration of the Examiner's rejection of claim 8.

Claims 9 and 10

Claims 9 and 10 depend on claim 8, which is allowable for the reasons given above. As a result, claims 9 and 10 are allowable as depending on an allowable independent claim.

Claim 11

In rejecting claim 11, the Examiner states that claim 11 is the same as claim 7 except claim 11 recites the method for communicating. Applicants respectfully assert that the Examiner has not addressed all of the limitations in claim 11. For example, the Examiner has not addressed the limitation that the active datastream includes a datastream identifier that identifies executable code for processing the active datastream. In addition, the Examiner has not addressed the limitation that the datastream class corresponds to a datastream identifier in the active datastream. Furthermore, the Examiner has not addressed the limitation of "the second computer system populating the instance of the datastream class with the active datastream received from the first computer system." Because the Examiner has failed to address these limitations in claim 11, the Examiner has failed to establish a prima facie case of anticipation for claim 11 under 35 U.S.C. §102(b).

Claim 11 is amended herein to specifically recite that the second computer system populates the instance of the datastream class with information contained in the datastream received from the first computer system. As stated above with respect to claims 1, 7, 8 and 16, the factory interface in De Borst creates an instance of the Stream class, but does not create an instance of a class corresponding to an identifier in a datastream received from a different computer system. For this reason, the factory interface in De Borst does not read on the step of constructing an active datastream in claim 11. Furthermore, nowhere does De Borst teach that a Stream object can populate itself with information contained in a datastream received from a different computer system by invoking at least one object method on the instance, as recited in claim 11. For this reason, the Stream object in De Borst does not read on the step of the second computer system populating the instance of the datastream class as recited in claim 11. Applicants assert that claim 11 is allowable over De Borst, and respectfully request reconsideration of the Examiner's rejection of claim 11.

Claims 12-14

Claims 12-14 depend on claim 11, which is allowable for the reasons given above. As a result, claims 12-14 are allowable as depending on an allowable independent claim.

Claim 15

In rejecting claim 15, the Examiner refers to the rejection of claims 1-4. In so doing, applicants respectfully assert that the Examiner has not addressed all of the limitations in claim 11. Nowhere in the rejection of claims 1-4 does the Examiner address the limitation of "the first computer system sending the active datastream to the second computer system by invoking a send method on the active datastream object." In addition, the Examiner has not addressed the limitation that the datastream class corresponds to a datastream identifier in the active datastream. Furthermore, the Examiner has not addressed the limitation of "the second computer system populating the new instance with the active datastream received from the first computer system." Because the Examiner has failed to address these limitations in claim 15, the Examiner has failed to establish a prima facie case of anticipation for claim 15 under 35 U.S.C. §102(b).

Claim 15 is amended herein to specifically recite that the second computer system populates the new instance with information contained in the active datastream received from the first computer system. As stated above with respect to claims 1, 7, 8, 11 and 16, the factory interface in De Borst creates an instance of the Stream class, but does not create an instance of a class corresponding to an identifier in a datastream received from a different computer system. For this reason, the factory interface in De Borst does not read on the second computer system creating an new instance of the datastream class in claim 11. Furthermore, nowhere does De Borst teach that a Stream object can populate itself with information contained in a datastream received from a different computer system by invoking at least one object method on the instance, as recited in claim 15. For this reason, the Stream object in De Borst does not read on the second computer system populating the new instance with information contained in the active datastream received from the first computer system, as recited in claim 15. Applicants assert that claim 15 is allowable over De Borst, and respectfully request reconsideration of the Examiner's rejection of claim 15.

Conclusion

In summary, De Borst does not teach, support, or suggest the unique combination of features in applicants' claims presently on file. Therefore, applicants respectfully assert that all of applicants' claims are allowable. Such allowance at an early date is respectfully requested. The Examiner is invited to telephone the undersigned if this would in any way advance the prosecution of this case.

Respectfully submitted,

MARTIN & ASSOCIATES, L.L.C.
P.O. Box 548
Carthage, MO 64836-0548
(417) 358-4700

By [illegible]
Derek P. Martin
Reg. No. 36,595

VERSION WITH MARKINGS TO SHOW CHANGES MADE

1. (Amended) A computer system comprising:

at least one processor;

a memory coupled to the at least one processor;

a datastream factory residing in the memory and executed by the at least one processor, the datastream factory creating an instance of a datastream class corresponding to an identifier <u>in a datastream</u> received from a second computer system; and

a datastream receive mechanism residing in the memory and executed by the at least one processor, the datastream receive mechanism populating the instance of the datastream class with [data] <u>information contained in the datastream</u> received from the second computer system by invoking at least one object method on the instance.

7. (Amended) A networked computer system comprising:

a first computer system coupled via a network connection to a second computer system;

each of the first and second computer systems comprising a datastream processor, the datastream processor including:

a datastream factory for creating an instance of an active datastream class corresponding to a datastream identifier received in a datastream on the network connection from the other computer system; and

a datastream receive mechanism that populates the instance of the active datastream class with [data] <u>information contained in the datastream</u> received on the network connection from the other computer system by invoking at least one object method on the instance.

8. (Amended) A networked computer system comprising:

a first computer system coupled via a network connection to a second computer system;

means for constructing an active datastream, the active datastream including a datastream identifier that identifies executable code for processing the active datastream;

means for sending the active datastream from the first computer system to the second computer system;

means for creating an instance of a datastream class that corresponds to the datastream identifier in the second computer system;

means for populating the instance of the datastream class with information contained in the active datastream received from the first computer system.

11. (Amended) A method for communicating between a first computer system and a second computer system, the method comprising the steps of:

the first computer system constructing an active datastream, the active datastream including a datastream identifier that identifies executable code for processing the active datastream;

the first computer system sending the active datastream to the second computer system;

the second computer system creating an instance of a datastream class that corresponds to the datastream identifier;

the second computer system populating the instance of the datastream class with information contained in the active datastream received from the first computer system by invoking at least one object method on the instance.

13. (Amended) The method of claim 11 wherein the step of populating the instance of the datastream with the information contained in the active datastream includes the step of executing a receive method on the instance of the datastream class.

15. (Amended) A method for communicating between a first computer system and a second computer system, the method comprising the steps of:

the first computer system constructing an active datastream object, the active datastream object including a datastream identifier that identifies a corresponding datastream class that includes executable code corresponding to a plurality of object methods for processing the active datastream object;

the first computer system sending the active datastream to the second computer system by invoking a send method on the active datastream object;

the second computer system reading the datastream identifier from the active datastream object received from the first computer system;

the second computer system creating a new instance of the datastream class that corresponds to the datastream identifier;

the second computer system populating the new instance with information contained in the active datastream received from the first computer system by invoking a receive method on the new instance; and

the second computer system performing a request represented by the active datastream by invoking at least one object method on the new instance.

16. (Amended) A program product comprising:

a datastream factory that creates an instance of a datastream class corresponding to [a received] an identifier in a received datastream;

a datastream receive mechanism that populates the instance of the datastream class with [received data] information contained in the received datastream by invoking at least one object method on the instance; and

signal bearing media bearing the datastream factory and the datastream receive mechanism.